# Center for Foundations of Intelligent Systems

Technical Report
97-07

Agent Based Velocity Control of
Highway Systems

W. KOHN, A. NERODE AND J. B. REMMEL

September 1997

## CORNELL
U N I V E R S I T Y

# REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

| 1. AGENCY USE ONLY ( Leave Blank) | 2. REPORT DATE 31 March 1998 | 3. REPORT TYPE AND DATES COVERED Technical Report |
|---|---|---|

**4. TITLE AND SUBTITLE**
Agent Based Velocity Control of Highway Systems

**5. FUNDING NUMBERS**
DAAH04-96-1-0341

**6. AUTHOR(S)**
W. Kohn, J.B. Remmel and A. Nerode

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Regents of the University of California
c/o Sponsored Projects Office
336 Sproul Hall
Berkeley, CA 94720-5940

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

ARO 35873.80-MA-MUR

**11. SUPPLEMENTARY NOTES**
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by the documentation.

**12 a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12 b. DISTRIBUTION CODE**

**13. ABSTRACT (Maximum 200 words)**
In this paper we present a hybrid systems approach to freeway traffic control. We introduce a model of traffic which includes both vehicles and voids, where a void is a unit of space greater than the normal following distance between vehicles. The introduction of voids allows us to successfully model traffic flow va a wave model. We discuss the fundamentals of the Multiple Agent Hybrid Control Architecture which was used to implement our control of freeway traffic and the results of a simulation of our control. the main conclusions of our simulation is that one can significantly increase thraffic throughput by controlling only 5-9% of the vehicles.

| 14. SUBJECT TERMS velocity control, highway systems, multiple agent hybrid control, simulation | | | 15. NUMBER OF PAGES 32 |
|---|---|---|---|
| | | | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED | 18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED | 20. LIMITATION OF ABSTRACT UL |
|---|---|---|---|

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

19980519 168

Technical Report
97-07

# Agent Based Velocity Control of Highway Systems

W. KOHN, A. NERODE AND J. B. REMMEL

September 1997

# Agent Based Velocity Control of Highway Systems

Wolf Kohn[*][1] and Anil Nerode[**][2] and Jeffrey B. Remmel[***][3]

[1] HyBrithms Corp.[†]
11201 S.E. 8th Street, Bldg. J, Suite 140
Bellevue, Washington
e-mail: wk@hybrithms.com
[2] Mathematical Sciences Institute
Cornell University, Ithaca, NY 14853
e-mail: anil@math.cornell.edu
[3] HyBrithms Corp. and
Department of Mathematics
University of California at San Diego San Diego, CA 92093
e-mail; jremmel@ucsd.edu

**Abstract.** In this paper we present a hybrid systems approach to freeway traffic control. We introduce a model of traffic which includes both vehicles and voids, where a void is a unit of space greater than the normal following distance between vehicles. The introduction of voids allows us to successfully model traffic flow via a wave model. We discuss the fundamentals of the Multiple Agent Hybrid Control Architecture which was used to implement our control of freeway traffic and the results of a simulation of our control. The main conclusion of our simulation is that one can significantly increase traffic throughput by controlling only 5-9% of the vehicles.

## 1   Introduction

This paper presents an architecture for the real-time feedback control of hybrid systems through a communication network composed of multiple decision makers herein refereed to as agents. The paper discusses some recent results from the theory of Hybrid systems, developed by the authors, related to the behavior of the architecture and illustrates them with a simple highway control system. This application was selected because it shares many elements of commonality with many other areas of application such as distributed control, sensor fusion, manufacturing shop-floor control and hybrid routing in communication networks; and yet it requires relatively modest modeling efforts.

Our architecture implements formal real-time intelligent controllers with learning capabilities. This architecture, termed the Multiple-Agent Hybrid Control Architecture (MAHCA), [21, 22], provides a knowledge-based, formal implementation

framework for deducing on-line feedback control and reactive strategies for processes involving multiple agents. The architecture includes capabilities for structural adaptation as a function of predictable and unpredictable events in the processes under control. This characteristic is necessary for satisfying performance requirements in the unavoidable presence of sensory and knowledge uncertainty.

We will illustrate the functional and operational characteristics of MAHCA in terms of a two-agent controller version whose goals are to maximize traffic throughput in a freeway network and to provide a dynamic route planning for selected cars through the network with minimum average time. The highway example was chosen because it explicitly exhibits some the basic properties of hybrid systems in a clear fashion. In addition we were able to demonstrate a property of these systems which is unique. That is, one can achieve a significant increase in throughput by controlling a relatively small number of vehicles. This result is a consequence of the implicit inter-vehicle constraints that limit the degrees of freedom for control in the freeway to a small percentage of the total vehicle by vehicle aggregated degrees of freedom. This implies that a platooning effect is achieved spontaneously by controlling the velocity of about 9-14% of the cars in the freeway network.

The organization of this paper is as follows. In section 2, we shall provide an overview of our freeway network model. In section 3, we shall descibe the generic variational model of MAHCA agent. In section 4, we shall give an overview of the architecture of MAHCA agent and how it carries out its computations. Finally in section 5, we will discuss the results of a simulation of our highway control. In particular, at the end of the paper, we will provide a series of charts which give the results of controlling 5%, 8%, and 9% of the vehicles.

## 2 Freeway Network Model and Control

The freeway network model we will use in our illustration of MAHCA is composed of two elements the Network Geometry and the Network Dynamics. We discus these elements in the next two subsections.

The substance in a traffic flow is, of course, vehicles. However, in addition to modeling the flow of vehicles, the model also includes the flow of voids, where a void is a unit of space greater than the normal following distance between vehicles. As vehicles (referred to generically as "cars" in what follows) travel from a source ($x = 0$) to the destination ($x = L$), the voids travel in the reverse direction, i.e. starting at $x = L$.

The introduction of voids eliminates a major problem with wave models. That is, such models are only valid for high uniform densities. Modeling the voids and their interaction with cars permits the generation of more realistic density distributions. When the voids are also modeled, the sparse density of vehicles become high densities of voids, allowing for model validity at low vehicle densities. This representation is patterned after the model of propagation of electrons and holes in semiconductor junctions.

For incorporating driver behavior and policies we use a Toda lattice representation vehicle interaction for each policy and generate the composite wave model via a formal aggregation procedure similar to the one proposed in [44].

A freeway segment is treated a single pipe with a capacity density corresponding to an average lane car velocity of 50 miles/hr for the number of lanes considered. Individual lanes are not modeled. Accidents are represented as reductions in the capacity of the segment.

## 2.1 Network Geometry

The network geometry is represented by a segment directed graph (SDG). An SDG is a structure composed of two types of sets: an edge set (E) and a node set (N). The set of edges (E), represent unidirectional freeways. Each edge (freeway) is modeled as a composite of one or more segments. The set of nodes (N) is composed of three, not necessarily disjoint, subsets: sources (S), interior nodes (I) and destination nodes (D). A source node represents a point at which cars enter the network, a destination node is a point at which cars leave the network and an interior node a point at which cars flow from one or more freeway segments to other freeway segments These concepts are illustrated in Figure 1.



**Fig. 1.** Geometry of the Model

From the geometric point of view, an edge is a connected straight line with a direction, indicated by an arrow at its destination point, containing two or more nodes. An edge is composed of one or more segments where a segment is a line in between two nodes. A two-way freeway is represented by two edges with opposite directions. Thus, independently of the actual map characteristics of a freeway segment, in our model a freeway segment is represented by a straight line which we assume has the same length as the freeway segment it represents. This simplifying assumption does

not cause inaccuracies because we model the constraints due to the geography and other constraints on traffic flow with a capacity function that we will describe in the next subsection.

Figure 2 shows the geometric model of the freeway system of the greater Houston area. This network was used to exercise our 2-agent implementation of MAHCA. Some of the simulation results will be presented in section 5.
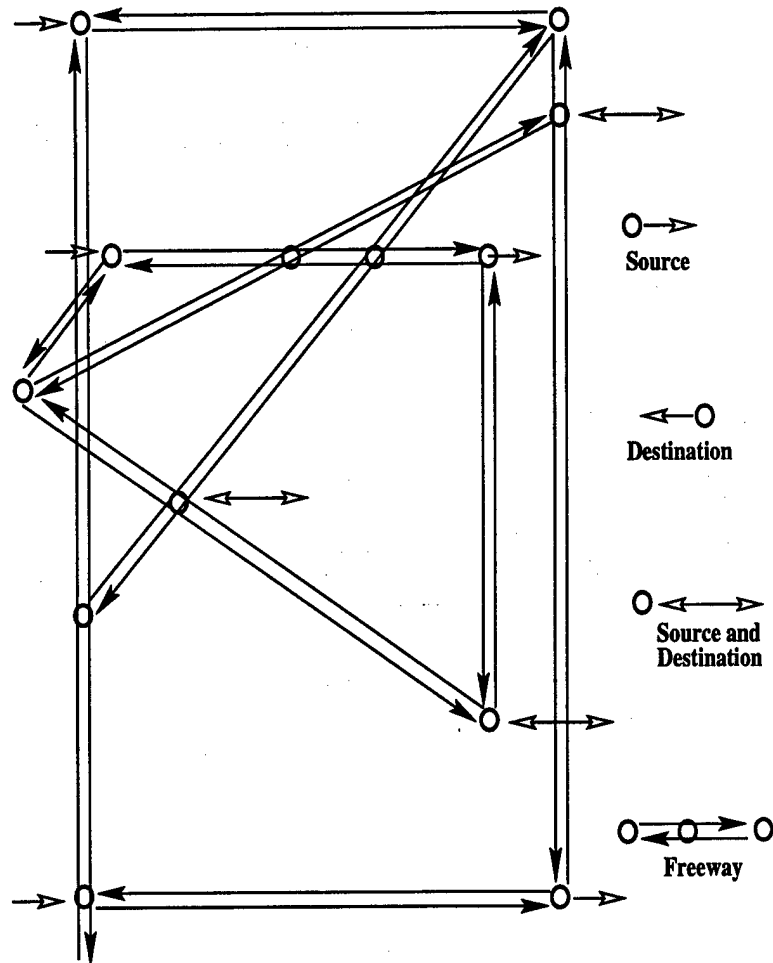


**Fig. 2.** Houston Highway Network

## 2.2 Freeway Network Dynamics

For the purpose of representing traffic flow dynamics, we view the freeway segments as pipes with variable cross-section carrying a composite fluid of two types of interacting particles, "car" and "void" particles. The void particles represent statistical

averages of inter-car distances. In each segment, cars flow in the direction of the segment edge while voids flows in the opposite direction.

The central principle behind our dynamics model of freeway networks is that the number of cars and voids flowing through a freeway segment are conserved quantities. This principle is an extension of the one proposed [16] and [24] and is now widely used in analysis of freeway control [25].

Rather than introduce the principle in its more general form, we will motivate it informally from a simple particle model of single lane freeway segments. This exercise is important because many aspects of freeway dynamics [24], such as driver behavior, are easily stated in terms of particle models.

The general idea behind particle models in a single lane freeway segment is the following. We set up a coordinate system with origin at the beginning node of the segment (we assume that a freeway segment inherits the direction of its edge). We assume that there are on the average $M$ cars and $M-1$ voids in the segment and that the speed at which the $n$-th car and the $n$-th void move, at time $t$ are functions of the positions of the $(n-1)$-th and $(n+1)$-th cars and the $(n-1)$-th void in the lane. In symbols,

$$\dot{x}_n(t+\delta) = f(s_{n-1}(t), x_{n-1}(t), x_{n+1}(t)) \qquad (1)$$

$$\dot{s}_n(t+\delta) = -g(s_{n-1}(t), x_{n-1}(t), x_{n+1}(t)) \qquad (2)$$

where $x_k(t)$ and $s_l(t)$ are respectively the position of the $k$-th car and $l$-th void in the segment at time $t$ and $f$ and $g$ are functions that capture driver behavior, car characteristics and geographic and environmental constraints. For the $n$-th car and $n$-th void ($n = 1, ..., M$) in the segment, the functions $f$ and $g$ map the positions of neighboring cars and voids at time $t$ to the speeds of the $n$-th car and $n$-th void at time $t + \delta$ where $\delta$ is a positive real number modeling average driver reaction time.

For the purposes of characterizing highway segment velocity control and car routing through the network, the model above is not satisfactory. We need a model that gives a global view of the dynamics. With this goal in mind we have transformed the particle model above into a wave model [25]. In a wave model, the cars, the voids and their interactions, in each freeway segment, are expressed by a set of coupled partial differential equations expressing conservation of mass (of cars and voids) and velocity of cars and voids. These equations are given in (3) - (5) and (6) and (7) below.

$$\frac{\partial \rho^c}{\partial t} + \frac{\partial Q^c}{\partial x} + k_1(C^c - \rho^c)\rho^v - k_2(C^v - \rho^v) = 0 \qquad (3)$$

with

$$Q^c = Q_0^c(\rho^c) - d\frac{\partial \rho^c}{\partial x}$$

$$\frac{\partial \rho^v}{\partial t} + \frac{\partial Q^v}{\partial x} + k_1(C^v - \rho^v)\rho^c - k_2(C^c - \rho^c) = 0 \qquad (4)$$

with

$$Q^v = Q_0^v(\rho^v) - d_1\frac{\partial \rho^v}{\partial x}$$

In (3) and (4), $t$ is time, $x$ is the freeway space variable, $\rho^j$, for $j = c$ or $v$, are the densities of the car and void particles (cars/mile, and voids /mile), $Q^j$, for $j = c$ or $v$, are the flows of cars and voids (cars /sec and voids/sec) respectively. The capacity of the freeway segment at each point is given by the functions $C^j$, for $j = c$ or $v$, in cars/sec and voids/sec respectively. The coefficients $k_1$ and $k_2$ are empirical coefficients expressing the transference from voids to cars and cars to voids respectively. Finally, $d$ and $d_1$ are aggregated empirical diffusion coefficients expressing driver behavior dependencies on the neighboring cars and voids. In (3) and (4), the functions $Q_0^j$, $j = c$ for cars or $j = v$ for voids, are called the free flows (for cars and voids respectively and measuring the number of cars and voids per second), and are given by

$$Q_0^j(\rho^j) = \frac{\sum_{i=1}^{n_j} a_i^j (\rho^j)^i}{\sum_{i=1}^{n_j} b_i^j (\rho^j)^i} \text{ with } j = c \text{ or } j = v \qquad (5)$$

where $a_i^j$ and $b_i^j$ are empirical coefficients. In particular for, Greenber's law of traffic flow for freeway segments under heavy load [24], (5) is the Padé approximant to the natural logarithmic function.

Equations (6) to (8) express the wave velocity dynamics $v^j$, for $j = c$ or $v$, of cars and voids along a freeway segment. In (6), $k_3$ and $V^u$ represent the control fields in each segment. $V^u$ is the recommended wave velocity and $k_3$ is the percentage of cars that are controlled. Since the dynamics of an individual car is constrained by its neighbors the controller does not have to control all the cars on the segment to achieve the desired *goal* which is to *maximize throughput* for the freeway network.

$$\frac{\partial v^c}{\partial t} + v^c \frac{\partial v^c}{\partial x} = -\frac{1}{\delta}\left(v^c - k_3 V^u + d\frac{\frac{\partial \rho^c}{\partial x}}{\rho^c}\right) \qquad (6)$$

$$\frac{\partial v^v}{\partial t} + v^v \frac{\partial v^v}{\partial x} = \frac{1}{\delta}\left(v^v - v^c + d_1\frac{\frac{\partial \rho^v}{\partial x}}{\rho^v}\right) \qquad (7)$$

The equations in (8), express the boundary conditions for each freeway segment. The right-hand side of the first equation expresses the gradient of the density of cars and voids being pumped into the segment. The right-hand side of the second equation in (8) expresses the density of particles leaving the segment.

$$\frac{\partial \rho^j}{\partial x}(t,0) = \rho^j(t)\ j = v \text{ or } c \qquad (8)$$
with
$$\rho^j(t,1) = e^j(t)$$

The exchange between voids and cars is given by

$$\frac{\partial \rho^c}{\partial t} + \frac{\partial \rho^v}{\partial t} + \frac{\partial Q_0^c}{\partial x} + \frac{\partial Q^v}{\partial x} = 0. \qquad (9)$$

This equation forms the rule which guarantees the conservation of the total number of "units" (cars and voids) in the freeway system.

The model outlined above, characterizes the dynamics of freeway segments and hence the freeway network. This model constitutes the specific (problem dependent)

equational knowledge required to control the freeway network using MAHCA. The next subsection formulates the highway velocity control problem as a multiple agent, knowledge-based control problem.

## 2.3    Control Problem

The overall goal of the control system is to maximize throughput for the freeway network by controlling the velocity $V^u$ and the percentage of controlled cars $k_3$, referred to as the control actions, in each segment. We associate with each segment i a control agent $A_i$ which generates the control actions for the corresponding segment as a function of sensory data, goal data, current status data and information from the other agents via a communication network herein referred to as the *Control Network*. A formal model of the control network and its dynamics is given in section 3.

# 3    Control Network

## 3.1    A MAHCA Agent's Model

In general, a hybrid system has a hybrid state, the simultaneous dynamical state of all plants and digital control devices. Properly construed, the hybrid states will form a differentiable manifold which we call the *carrier manifold* of the system. To incorporate the digital states as certain coordinates of points of the carrier manifold, we "continualize" the digital states. That is, we view the digital states as finite, real-valued, piecewise-constant functions of continuous time and then we take smooth approximations to them. This also allows us to consider logical and differential or variational constraints on the same footing, each restricting the points allowed on the carrier manifold. In fact, all logical or discontinuous features can be continualized without practical side-effects. This is physically correct since for any semiconductor chip used in an analog device, the zeros and ones are really just abbreviations for sensor readings of the continuous state of the chip. Every constraint of the system, digital or continuous, is incorporated into the definition of what points are on the carrier manifold. Lagrange constraints are regarded as part of the definition of the manifold as well, being restrictions on what points are on the manifold.

More specifically, let $A_i$, $i = 1, \ldots, N(t)$ denote the agents active at the current time $t$ . In our model, $t$ takes values on the real line. At each time $t$, the status of each agent in the network is given by a point in a locally differentiable manifold $M$ [24]. The Behavior function $B_i$ of an active agent $A_i$ is given by a continuous function,

$$B_i : M \times T \to R^+ \tag{10}$$

where $T$ is the real line (time space) and $R^+$ is the positive real line. $M$ is contained in the Cartesian Product

$$M \subseteq G \times S \times X \times A \tag{11}$$

where $G$ is the spaces of goals, $S$ is the space of sensory data, $X$ is the space of controller states and $A$ is the space of control actions. In the freeway network, $X$ is the space of current distributions of car and void densities and velocities in

its segments, $S$ is the space of measurements of car and void densities along the segment (one sensor each third of a mile), $A$ is the space spanned by the velocity and percentage fields, and $G$ is the set of car and void densities that maximizes throughput.

From an agent's point of view, the dynamics of the control network is characterized by certain trajectories on the manifold $M$. These trajectories characterize the flow of information through the network and its status. Specifically, we need to define two items:

(i) a set of generators for the behavior functions at time $t$,

$$\{B_i(p,t) : i = 1,\ldots N, p \in M\} \tag{12}$$

and

(ii) the control actions issued by the agents.

We will see shortly that these actions are implemented as infinitesimal transformations defined in $M$. The general structure of the behavior function in (12) for an agent $A_i$ at time $t$ is given in (13) below:

$$B_i(p,t) = F_i(U_i, B, \alpha_i)(p,t) \tag{13}$$

where $F_i$ is a smooth function, $B$ is the vector of behavior functions, $C_i^u$ is the behavior modification function for the $i$-th agent, and $\alpha_i$ is the command action issued by the $i$-th agent. We will devote the rest of this subsection to characterizing this model.

We start with a discussion of the main characteristics of the manifold $M$. In general a manifold $M$ is a topological space (with topology $\Theta$) composed of three items:

(a) A set of points of the form of (11).
(b) A countable family of open subsets of M, $U_i$ such that

$$\bigcup_i U_i = M.$$

(c) A family of smooth homeomorphisms, $\{\phi_i : \phi_i : U_i \to V_i\}$, where for each $j$, $V_j$ is an open set in $R^k$. The sets $U_i$ are referred to in the literature as coordinate neighborhoods or charts. For each chart $U_j$ the corresponding function $\phi_j$ is referred to as its coordinate chart.

The coordinate chart functions satisfy the following additional condition:

Given any charts $U_i$ and $U_j$ such that $U_i \cap U_j \neq \emptyset$, the function $\phi_i \circ \phi_j^{-1} :$ $\phi_j(U_i \cap U_j) \to \phi_i(U_i \cap U_j)$ is smooth.

In the literature, one usually finds an additional property, which is the Hausdorff property in the definition of manifolds [25]. Since this property does not hold in our application we will not discuss it.

Now we proceed to customize the generic definition of the manifold to our application. We start with the topology $\Theta$ associated with $M$. We note that the points

of $M$ have a definite structure, see (11), whose structure is characterized by the interval of values in the space $G$ of goals, the space $S$ of sensory data, the space $X$ of controller states and the space $A$ of control actions. The number of these parameters equals $k$. The knowledge about these parameters is incorporated into the model by defining a finite topology $\Omega$ on $R^k$ [5].

The open sets in $\Omega$ are constructed from the clauses encoding what we know about the parameters. The topology $\Theta$ of $M$ is defined in terms of $\Omega$ as follows. For each open set $W$ in $\Omega$ such that $W \subseteq V_j \subseteq R^k$, we require that the set $\phi_j^{-1}(W)$ be in $\Theta$. The sets constructed in this way form a basis for $\Theta$ so that a set $U \subseteq M$ is open if and only if for each $p \in U$, there is $j$ and an open set $W \in \Omega$ such that $W \subseteq V_j$ and $p \in \phi_j^{-1}(W)$.

To characterize the actions commanded by a MAHCA agent we need to introduce the concept of derivations on $M$. Let $F_p$ be the space of real valued smooth functions $f$ defined in a neighborhood a point $p$ in $M$. Let $f$ and $g$ be functions in $F_p$. A *derivation* $v$ of $F_p$ is a map

$$v : F_p \to F_p$$

that satisfies the following two properties.

$$v(f + g)(p) = (v(f) + v(g))(p) \qquad \text{(Linearity)} \qquad (14)$$

$$v(f \cdot g)(p) = (v(f) \cdot g + f \cdot v(g))(p) \qquad \text{( Leibniz Rule)} \qquad (15)$$

Derivations define vector fields on $M$ and a class of associated curves called integral curves [26]. Suppose that $C$ is a smooth curve on $M$ parameterized by $\psi : I \to M$ where $I$ a subinterval of $R$. In local coordinates, $p = (p^1, ..., p^k)$, $C$ is given by $k$ smooth functions $\psi(t) = (\psi^1(t), \dots, \psi^k(t))$ whose derivative with respect to t is denoted by $\dot\psi(t) = (\dot\psi^1(t), \dots, \dot\psi^k(t))$. We introduce an equivalence relation on curves in $M$ as the basis of the definition of tangent vectors at a point in $M$ [14]. Two curves $\psi_1(t)$ and $\psi_2(t)$ passing through a point $p$ are said to be equivalent at $p$ (notation: $\psi_1(t) \sim \psi_2(t)$), if there exists $\tau_1, \tau_2 \in I$ such that

$$\psi_1(\tau_1) = \psi_2(\tau_2) = p \qquad (16)$$

$$\dot\psi_1(\tau_1) = \dot\psi_2(\tau_2). \qquad (17)$$

Clearly, $\sim$ defines an equivalence relation on the class of curves in $M$ passing through $p$. Let $[\psi]$ be the equivalence class containing $\psi$. A *tangent vector* to $[\psi]$ is a derivation, $v|_p$, such that in local coordinates $(p^1, \dots, p^k)$, it satisfies the condition that given any smooth function $f : M \to R$,

$$v|_p(f)(p) = \sum_{j=0}^{k} \psi^j(t) \frac{\partial f(p)}{\partial p^j} \qquad (18)$$

where $p = \psi(t)$. The set of tangent vectors associated with all the equivalence classes at $p$ defines a vector space called the *tangent vector space* at $p$, denoted by $TM_p$. The set of tangent spaces associated with points in $M$ can be "glued" together to form a manifold called the *tangent bundle* which is denoted by $TM$,

$$TM = \bigcup_{p \in m} TM_p.$$

For our purposes, it is important to specify explicitly how this gluing is implemented. This will be explained below after we introduce the concept of a vector field and discuss its relevance in the model.

A *vector field* on $M$ is an assignment of a derivation $v|_p$ to each point $p$ of $M$ which varies smoothly from point to point. That is, if $p = (p^1, ..., p^k)$ are local coordinates, then we can always write $v|_p$ in the form

$$v|_p = \sum_j \lambda^j(p) \frac{\partial}{\partial p^j} \tag{19}$$

Then $v$ is a vector field if the coordinate functions $\lambda_i$ are smooth.

Comparing (18) and (19) we see that if $\psi$ is a parameterized curve in $M$ whose tangent vector at any point coincides with the value of $v$ at a point $p = \psi(t)$, then in the local coordinates $p = (\psi^1(t), \ldots, \psi^k(t))$, we must have

$$\dot{\psi}^j(t) = \lambda^j(p) \text{ for } j = 1, \ldots, k. \tag{20}$$

In our application, each command issued by the MAHCA agent is implemented as a vector field in $M$. Each agent constructs its command field as a combination of 'primitive' predefined vector fields. Since the chosen topology for $M$, $\Theta$, is not metrizable, we cannot guarantee a unique solution to (20) in the classical sense for a given initial condition. However, they have solutions in a class of continuous trajectories in $M$ called *relaxed curves* [33]. In this class, the solutions to (20) are unique. We discuss the basic characteristics of relaxed curves as they apply to our process control formulation and implementation in Section 4. Next, we describe some of their properties as they relate to our plant model and control process. For this objective, we need to introduce the concept of flows in $M$.

If $v$ is a vector field, any parameterized curve passing through a point $p$ in $M$ is called an *integral curve associated with $v$* if in local coordinates (5) holds. An integral curve associated with a field v, denoted by $\Psi(t, p)$ is termed the flow generated by $v$ if it satisfies the following properties:

$$\Psi(t, \Psi(\tau, p)) = \Psi(t + \tau, p) \text{ (semigroup property)} \tag{21}$$
$$\Psi(0, p) = p \text{ (initial condition)}$$
$$\text{and}$$
$$\frac{d}{dt}\Psi(t, p) = v |_{\Psi(t,p)} \text{ (flow generation)}$$

Now we are ready to customize these concepts for our model. Let $\Delta > 0$ be the width of the current decision interval, $[t, t + \Delta)$. Let $C_i^u(p, t)$ be the unsatisfied demand at the beginning of the interval. Agent $A_i$ has a set of primitive actions:

$$\{v_{i,j} : j = 1, \ldots, n_i \text{ where } v_{i,j}|_p \in TM_p \text{ for each } p \in m\}. \tag{22}$$

During the interval $[t, t + \Delta)$, agent $A_i$ schedules one or more of these actions to produce a flow which will bring the system closer to the goal. In particular, $A_i$ determines the fraction $\alpha_{i,j}(p, t)$ of $\Delta$ that action $v_{i,j}$ must be executed as a function of the external perturbation functions $S_{r,i}(t, p)$ and the vector of the behavior functions of the agents in the network $B(p, t) = (B_1(p, t), \ldots, B_{N(t)})$. Figure 3 conceptually
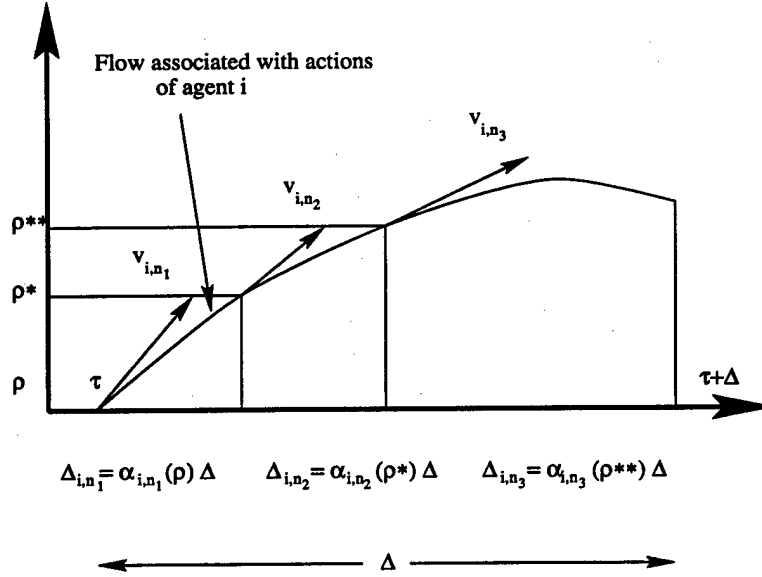
**Fig. 3.** Conceptual illustration of agent action schedule

illustrates a schedule of actions involving three primitives. We will use this example as means for describing the derivation of our model. The general case is similar.

The flow $\Psi_i$ associated with the schedule of Figure 3 can be computed from the flows associated with each of the actions:

$$\Psi_i(\tau,p) = \begin{cases} \Psi_{v_{i,n_1}}(\tau,p) \text{ if } t \le \tau \le t + \Delta_{i,n_1} \\ \Psi_{v_{i,n_2}}(\tau,\Psi_{v_{i,n_1}}(\tau,p)) \text{ if } t + \Delta_{i,n_1} \le \tau \le t + \Delta_{i,n_1} + \Delta_{i,n_2} \\ \Psi_{v_{i,n_3}}(\tau,\Psi_{v_{i,n_2}}(\tau,\Psi_{v_{i,n_1}}(\tau,p))) \\ \qquad \text{if } t + \Delta_{i,n_1} + \Delta_{i,n_2} \le \tau \le t + \Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3} \end{cases} \qquad (23)$$

where $\Delta = \Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3}$ and $\alpha_{i,n_1} + \alpha_{i,n_2} + \alpha_{i,n_3} = 1$. We note that the flow $\Psi_i$ given by (23) characterizes the evolution of the process as viewed by agent $A_i$. The vector field $v_i|_p$ associated with the flow $\Psi_i$ is obtained by differentiation and the third identity in (21). This vector field applied at $p$ is proportional to

$$[v_{i,n_1}, [v_{i,n_2}, v_{i,n_3}]] \qquad (24)$$

where $[.,.]$ is the Lie bracket due to the parallelogram law, see [42]. The Lie bracket is defined as follows. Let $v$ and $w$, be derivations on $M$ and let $f : M \to R$ be any real valued smooth function. The Lie bracket of $v$ and $w$ is the derivation defined by

$$[v,w](f) = v(w(f)) - w(v(f)),$$

see [10].

Thus the composite action $v_i|_p$ generated by the $i$-th agent to control the process is a composition of the form of (24). Moreover from a version of the Chattering

lemma and duality [18], we can show that this action can be expressed as a linear combination of the primitive actions available to the agent as follows.

$$[v_{i,n_1}, [v_{i,n_2}, v_{i,n_3}]] = \sum_j \gamma_j^i(\alpha) v_{i,j} \tag{25}$$

$$\sum_j \gamma_j^i(\alpha) = 1$$

where the coefficients $\gamma_j^i$ determined by the fraction of time that each primitive action $v_{i,j}$ is used by agent i.

The effect of the field defined by the composite action $v_i|_p$ on any smooth function (equivalent function class) is computed by expanding the right hand side of (23) in a Lie-Taylor series [10]. In particular, we can express the change in the behavior modification functions $C_i^u$ due to the flow over the interval $\Delta$ in terms of $v_i|_p$. The evolution of the modified behavior function $C_i^u$ over the interval starting at point $p$ is given by

$$C_i^u(t + \Delta, p'') = C_i^u(t, \Psi_i(t + \Delta, p)) \tag{26}$$

Expanding the right hand side of (26) in a Lie-Taylor series around $(t, p)$, we obtain,

$$C_i^u(t + \Delta, p'') = \sum_j \frac{(v_i|_p(C_i^u(p, t)))^j \Delta^j}{j!}$$

where

$$(v_i|_p(\cdot))^j = v_i|_p\left((v_i|_p(\cdot))^{j-1}\right) \tag{27}$$

and

$$(v_i|_p)^0(f) = f \text{ for all } f.$$

In general, the series in the right hand side of (27) will have countable many non-zero terms. In our case, since the topology of $M$ is finite due to the fact that it is generated by finitely many logic clauses, this series will have only finitely many non-zero terms. Intuitively, this is so because in computing powers of derivations (i.e., limits of differences), we need only to distinguish among different neighboring points. In our formulation of the topology of $M$, this can only be imposed by the information in the clauses of the agent's Knowledge Base. Since each agent's Knowledge Base has only finitely many clauses, there is a term in the expansion of the series in which the power of the derivation goes to zero because we cannot distinguish between two points which cannot be separated by open sets in the finite topology. This is important because it allows the series in the right-hand side to be effectively generated by a locally finite automaton. We will expand on the construction of this automaton in the next section when we discuss the inference procedure carried out by each agent.

We note that given the set of primitive actions available to each agent, the composite action is determined by the vector of fraction functions $\alpha_j$ . We will see in the next section that this vector is inferred by each agent from the proof of existence of solutions of an optimization problem.

Now we can write the specific nature of the model formulated in expression (13). At time $t$ and at point $p \in M$ the behavior modification function of agent $i$ is given by :

$$C_i^u(p,t) = C_i^u(p,t^-) + S_{r,i}(p,t) + \sum_k Q_{i,k}B_k(p,t^-) \qquad (28)$$

where $t^-$ is the end point of the previous update interval, $S_{r,i}$ is the external perturbation function to agent $i$, and $Q_{i,k}$ is a multiplier determining how much of the current behavior modification requirements of agent $A_k$ is allocated to agent $A_i$. This allocation is determined from the characteristics of the process both agents are controlling and from the process description encoded in the agent's knowledge base. The actual request for service from agent $k$ to agent $i$ is thus the term, $Q_{i,k}B_k(p,t^-)$. The information sent to agent $i$ by agent $k$ is the behavior modification function $B_k(p,t^-)$ at the end of the previous interval. Finally the point $p \in M$ carries the current status of the process monitored by the agents appearing in (28). Agent $k$ thus contributes to Agent $i$'s new control only if $Q_{i,k} \neq 0$.

This concludes our description of the model. For space considerations, some details have been left out. In particular those related to the strategy for activation and deactivation of agents. These will be discussed in a future paper.

## 4    The Multiple Agent Hybrid Control Architecture

In this section, we describe the main operational and functional characteristics of our intelligent controller. As we mentioned in the introduction, this controller is implemented as a distributed system composed of agents and a communication network. We referred to the latter as the control network in the previous section. The architecture realizing this system, called the Multiple-Agent Hybrid Control Architecture (MAHCA), operates as an on-line distributed theorem prover. At any update time, each active agent generates actions as side effects of proving an existentially quantified subtheorem (lemma) which encodes the desired behavior of the logic communications network as viewed by the agent. The conjunction of lemmas at each instant of time, encodes the desired behavior of the entire network.

Each agent of MAHCA consists of five modules: a Planner, a Dynamic Knowledge Base, a Deductive Inferencer, an Adapter and a Knowledge Decoder. We briefly overview the functionality of an agent in terms of its modules.

The *Knowledge Base* stores the requirements of operations or processes controlled by the agent. It also encodes system constraints, inter-agent protocols and constraints, sensory data, operational and logic principles and a set of primitive inference operations defined in the domain of equational terms.

The *Planner* generates a statement representing the desired behavior of the system as an existentially quantified logic expression herein referred to as the Behavior Statement.

The *Inferencer* determines whether this statement is a theorem in the theory currently active in the Knowledge Base. If the behavior statement logically follows from the current status of the knowledge base, the inferencer generates, as a side effect of proving this behavior statement to be true, the current control action schedule. If the behavior statement does not logically follow from the current status of

the knowledge base, that is, the desired behavior is not realizable, the inferencer transmits the failed terms to the *Adapter* module for replacement or modification.

Finally, the *Knowledge Decoder* translates data from the other agents and incorporates them into the Knowledge Base of the agent.

In each agent of MAHCA, the behavior statement is the formulation of a relaxed variational optimization problem whose successful resolution produces an action schedule of the form of (24). Each agent operates as a real-time theorem prover in the domain of relaxed variational theory [39]. A customized version of this theory, enriched with elements of differential geometry, equational logic and automata theory provides a general representation for the dynamics, constraints, requirements and logic of the control network . We devote the rest of this section to the discussion of the main elements of this theory in the context of the operational features of MACHA.

The architecture is composed of two items: The Control Agent, and the control network. These items are illustrated in Figures 4 and 5 respectively. We will discuss them in the remaining of this section.

## 4.1 Architectural Elements of a control agent:

We will discuss next the functionality of the five modules of a control agent. These are: the Knowledge Base, the Planner, the Inferencer, the Knowledge Decoder and the Adapter.

**4.1.1. Knowledge Base:** The Knowledge Base consists of a set of equational first order logic clauses with second order extensions. The syntax of clauses is similar to the ones in the Prolog language. Each clause is of the form

$$Head \leftarrow Body \tag{29}$$

where *Head* is a functional form, $p(x_1, ... x_n)$, taking values in the binary set $[true, false]$ and $x_1, x_2, \ldots, x_n$ are variables or parameters in the domain $M$ of the MAHCA network. The symbol $\leftarrow$ stands for logical implication. The variables appearing in the clause head are assumed to be universally quantified. The *Body* of a clause is a conjunction of one or more logical terms

$$e_1 \wedge e_2 \wedge \ldots \wedge e_n \tag{30}$$

where $\wedge$ is the logical 'and'. Each term in (30) is a relational form. A relational form is one of the following: an equational form, an inequational form, a covering form, or a clause head. The logical value of each of these forms is either true or false. A relational form $e_i$ is true precisely at the set of tuples of values $S_i$ of the domain taken by the variables where the relational form is satisfied and is false for the complement of that set. Thus for $e_i = e_i(x_1, \ldots, x_n)$, $S_i$ is the possibly empty subset of $M^n$

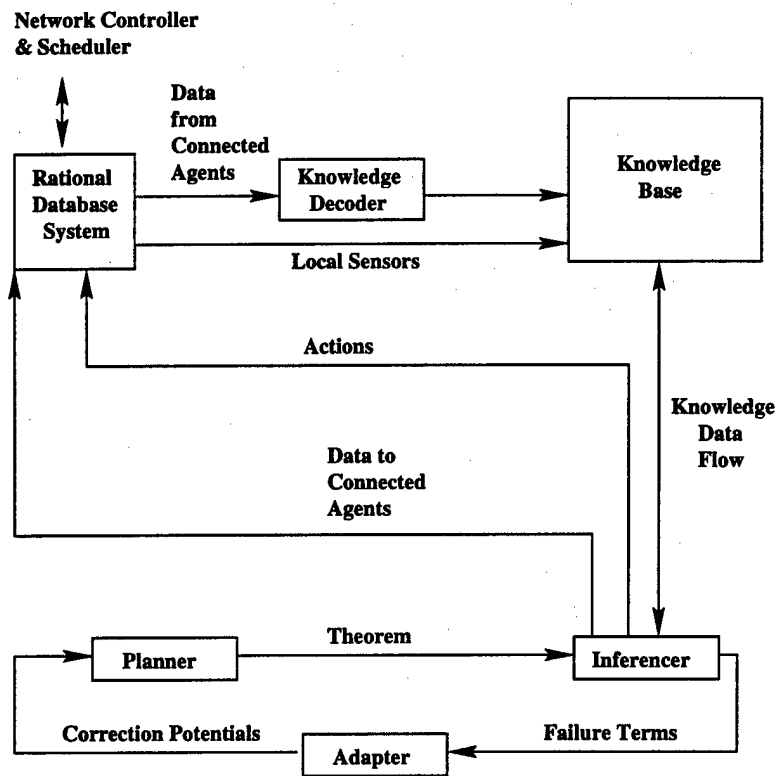$$S_i = \{(x_1, \ldots, x_n) \in M^n : e_i(x_1, \ldots, x_n) = true\}$$

**Fig. 4.** Control Agent



**Declartive Agent**
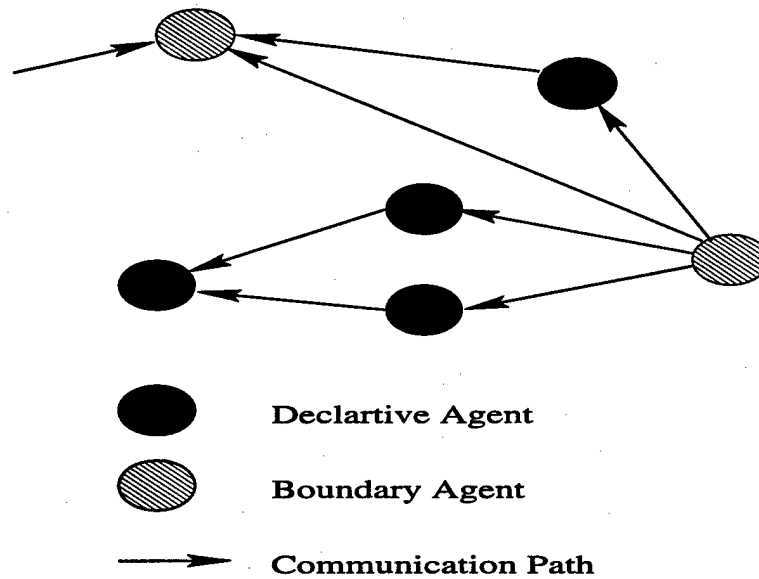
**Boundary Agent**

**Communication Path**

**Fig. 5.** Network of Cooperating Control Agents

so that

$$e_i(x_1, \ldots, x_n) = false \text{ if } (x_1, \ldots, x_n) \in M^n/S_i.$$

The generic structure of a relational form is given in Table 1.

| Form | Structure | Meaning |
|---|---|---|
| equational | $w(x_1, \ldots, x_n) = v(x_1, \ldots, x_n)$ | equal |
| inequational | $w(x_1, \ldots, x_n) \neq v(x_1, \ldots, x_n)$ | not equal |
| covering | $w(x_1, \ldots, x_n) < v(x_1, \ldots, x_n)$ | partial order |
| clause head | $q(x_1, \ldots, x_n)$ | recursion,chaining |

Table 1. Structure of the Relational Form

In Table 1, $w$ and $v$ are polynomic forms with respect to a finite set of operations whose definitional and property axioms are included in the Knowledge Base. A polynomic form $v$ is an object of the form $v(x_1, \ldots, x_n) = \sum_{\omega \in \Omega}(v, \omega) \cdot \omega$ where $\Omega^*$ is the free monoid generated by the variable symbols $\{x_1, \ldots, x_n\}$ under juxtaposition. The term $(v, \omega)$ is called the coefficient of $v$ at $\omega$. The coefficients of a polynomic form $v$ take values in the domain of definition of the clauses. The domain in which the variables in a clause head take values is the manifold $M$ described in section 2. The logical interpretation of (29) and (30) is that the *Head* is true if the conjunction of the terms of *Body* are jointly true for instances of the variables in the clause head. $M$ is contained in the Cartesian product :

$$M \subseteq G \times S \times X \times A \tag{31}$$

where $G$ is the space of goals, $S$ is the space of sensory data, $X$ is the space of controller states and $A$ is the space of control actions. These were described in section 2. $G$, $S$, $X$, and $A$ are manifolds themselves whose topological structure are defined by the specification clauses in the Knowledge Base, see Figure 6. These clauses, which are application dependent, encode the requirements on the closed-loop behavior of the model of the agent. In fact the closed loop behavior, which we will define later in this section in terms of a variational formulation, is characterized by continuous curves with values in $M$. This continuity condition is central because it is equivalent to requiring the system to look for actions that make the closed loop behavior satisfy the requirements of the plant model.

The denotational semantics of each clause in the knowledge base is one of the following:

1. a conservation principle,
2. an invariance principle, or
3. a constraint principle.

*Conservation principles* are one or more clauses about the balance of a particular process in the dynamics of the system or the computational resources. For instance, equation (28) encoded as a clause expresses the conservation of demand in the logic communications network

$$conservation\_of\_unsatisfied\_demand(p,t,[Q_{i,k}],S_{r,i},[B_k],\Delta,C_i^u(t,p)) \leftarrow$$

$$C_i^u(t+2\Delta,p'')=\sum_j \frac{\left(v_i|_p(C_i^u(t+\Delta,p'))\right)\Delta^j}{j!} \wedge$$

/* encoding of equation (13) */

$$C_i^u(t+\Delta,p')=C_i^u(t,p)+S_{r,i}(t,p)+\sum_k Q_{i,k}B_k(t,p,\dot{p}) \wedge \qquad (32)$$

/* encoding of equation (15) */

$$process\_evolution(p,t,p'') \wedge \quad \text{/* encoding of equation (13) */}$$

$$consevation\_of\_unsatisfied\_demand(p'',t+\Delta,[Q_{i,k}],S_{r,i},[B_k],\Delta,C_i^u(t+2\Delta,p''))$$

In (32), the first equational term relates the segment car density for agent $i$ at the current time to the density in the past and the net current density of the other agents connected to agent $i$. The last term of the rule implements the recursion.

As another example, consider the following clause representing conservation of computational resources:

$$comp(Load,Process,Op\_count,Limit) \leftarrow process(process\_count)$$

$$\wedge\ process\_count \cdot Load_1 - Op\_count < Load$$

$$\wedge\ Load_1 < Limit$$

$$\wedge\ comp(Load_1,Process,Op\_count,Limit).$$

where *Load* corresponds to the current computational burden, measured in VIPS (Variable Instantiations Per Second), *Process* is a clause considered for execution, and *Op_count* is the current number of terms in process.

Conservation principles always involve recursion whose scope is not necessarily a single clause, as in the example above, but with chaining throughout several clauses.

*Invariance principles* are one or more clauses establishing constants of the evolution of agent's behavior modification functions in a general sense. These principles include stationarity, which plays a pivotal role in the formulation of the theorems proved by the architecture, and geodesics. The necessary conditions for maximizing throughput in the freeway network constitute an example of this type of principle. The importance of invariance principles lies in the reference they provide for the direction of unexpected events.

*Constraint principles* are clauses representing engineering limits to actuators or sensors and, most importantly, behavior policies. The clauses defining the capacity function in each segment are examples of these type of principles. Another example in this application is given by the clauses that define the lifting strategy for embedding discrete varying trajectories into $M$ (interpolation rules).

The clause database is organized in a nested hierarchical structure illustrated in Figure 6. The bottom of this hierarchy contains the equations that characterize the algebraic structure defining the terms of relational forms, i.e an algebraic variety [45].

At the next level of the hierarchy, three types of clauses are stored: Generic Control Specifications, System Representation and Goal Class Representation.
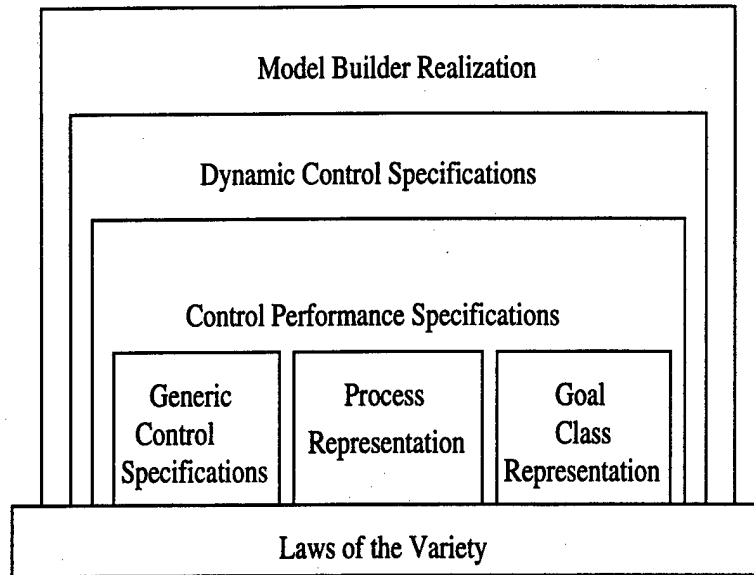
```
┌─────────────────────────────────────────────────────┐
│                Model Builder Realization             │
│  ┌───────────────────────────────────────────────┐  │
│  │          Dynamic Control Specifications        │  │
│  │  ┌─────────────────────────────────────────┐  │  │
│  │  │      Control Performance Specifications   │  │  │
│  │  │  ┌─────────┐ ┌─────────┐ ┌─────────┐     │  │  │
│  │  │  │ Generic │ │ Process │ │  Goal   │     │  │  │
│  │  │  │ Control │ │ Repres. │ │  Class  │     │  │  │
│  │  │  │ Specs   │ │         │ │ Repres. │     │  │  │
└──┴──┴──┴─────────┴─┴─────────┴─┴─────────┴─────┴──┴──┘
│                  Laws of the Variety                 │
└──────────────────────────────────────────────────────┘
```

**Fig. 6.** Knowledge Base Organization

The *Generic Control Specifications* are clauses expressing general desired behavior of the system. They include statements about stability, complexity and robustness that are generic to the class of declarative rational controllers. These specifications are written by constructing clauses that combine laws of the kind which use the Horn clause format described earlier. Examples of these types of clausse are the one that specifies the range of the parameters in the traffic model for which the system is stable.

The *Process Representation* is given by clauses characterizing the dynamic behavior and structure of the plant, which includes sensors and actuators. These clauses are written as conservation principles for the dynamic behavior and as invariance principles for the structure. As in the case of Generic Control Specifications, they are constructed by combining a variety of laws in the equational Horn clause format. An example of this type of clause are the density and velocity equations of the freeway segments given in section 2 encoded in Horn clause format.

The *Goal Class Representation* contains clauses characterizing sets of desirable operation points in the domain (points in the manifold $M$). These clauses are expressed as soft constraints; that is, constraints that can be violated for finite intervals of time. They express the ultimate purpose of the controller but not its behavior over time.

The next level of the hierarchy involves the *Control Performance Specifications*. These are typically problem dependent criteria and constraints. They are written in equational Horn clause format. They include generic constraints such as speed and time of response, and qualitative properties of state trajectories [39].

*Dynamic Control Specifications* are equational Horn clauses whose bodies are modified as a function of the sensor and goal commands.

Finally, *Model Builder Realization* clauses constitute a recipe for building a procedural model (an automaton,) for generating variable instantiation (unification), and for theorem proving.


### 4.1.2. The Planner:

The function of the theorem Planner, which is domain-specific, is to generate, for each update interval, a symbolic statement of the desired behavior of the system, as viewed, say by agent $j$, throughout the interval. The theorem statement that it generates has the following form.

Given a set of primitive actions there exists a control schedule $v_i|_p$ of the form (25) and a fraction function differential $d\alpha(\cdot)$ (Figure 3) in the control interval $[t, t + \Delta)$ such that $d\alpha(\cdot)$ minimizes the functional

$$\int_t^{t+\Delta} L_i\big(\Psi_i(\tau, p), v_i|_p(G(_i(\tau, p)))\big) d\alpha(p, d\tau) \tag{33}$$

subject to the following constraints:

$$g_i(S_i, \Psi_i(t + \Delta, p)) = G_i(t, X_i)$$

(local goal for the interval),

$$\sum_m Q_{i,m}(p, t) L_m(p, t) = V_i(p, t) \tag{34}$$

(inter-agent constraint, see (28))

and

$$\int_t^{t+\Delta} d\alpha(p, d\tau) = 1$$

In (33), $L_i$ is the *Local Relaxed Lagrangian* of the system as viewed by Agent $i$ for the current interval of control $[t, t+\Delta)$. This function, which maps the Cartesian product of the state and control spaces into the real line with the topology defined by the clauses in the knowledge base, captures the dynamics, constraints and requirements of the system as viewed by agent $i$. The relaxed Lagrangian function $L_i$ is a continuous projection in the topology defined by the knowledge base (see [36]) in the coordinates of the $i$-th agent of the global Lagrangian function $L$ that characterizes the system as a whole.

In (34), $p$ represents the state of the process under control as viewed by the agent and $G_i$ is the parallel transport operator bringing the goal to the current interval, see [27]. The operator $G_i$ is constructed by lifting to the manifold the composite flow (see equation (23)). We note that the composite flow and the action schedule are determined once the fraction function is known and that this function is the result of the optimization (33), (34). In particular, the action schedule is constructed as a linear combination of primitive actions, see equation (25).

The term $d\alpha(\cdot)$ in (33) is a Radon probability measure [40] on the set of primitive control actions or derivations that the agent can execute for the interval $[t, t + \Delta)$. It measures, for the interval, the percentage of time to be spent in each of the primitive
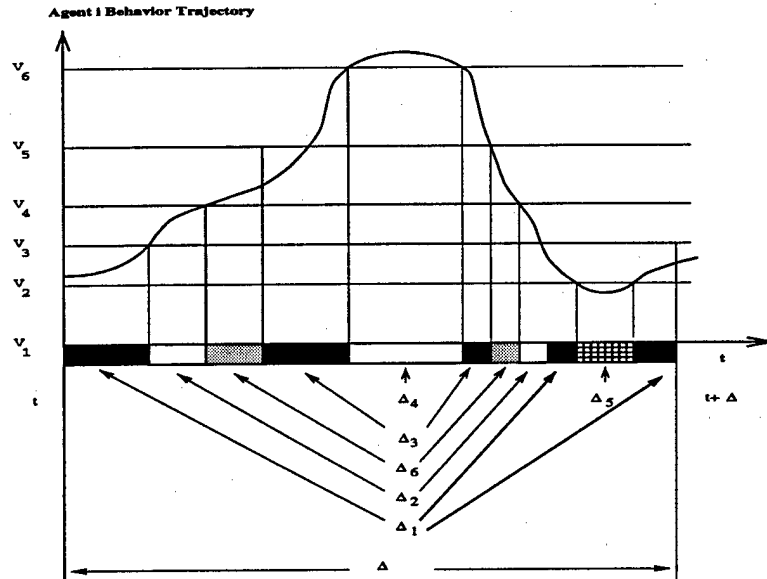
**Fig. 7.** Illustration of optimization

derivations. The central function of the control agent is to determine this mixture of actions for each control interval. This function is carried out by each agent by inferring from the current status of the knowledge base whether a solution of the optimization problem stated by the current theorem exists, and, if so, to generate corresponding actions and state updates. Figure 7 illustrates the relations between the primitive actions and the fraction of $\Delta$ they are active in the interval $[t, t + \Delta)$.

The expressions in (34) constitute the constraints imposed in the relaxed optimization problem solved by the agent. The first one is the local goal constraint expressing the general value of the state at the end of the current interval. The second represents the constraints imposed on the agent by the other agents in the network. Finally, the third one indicates that is a probability measure. Under relaxation and with the appropriate selection of the domain, see [21], the optimization problem stated in (33) and (34) is a convex optimization problem. This is important because it guarantees that if a solution exists, it is unique up to probability, and also, it guarantees the computational effectiveness of the inference method that the agent uses for proving the theorem.

The construction of the theorem statement given by (33) and (34) is the central task carried out in the Planner. It characterizes the desired behavior of the process as viewed by the agent in the current interval so that its requirements are satisfied and the system "moves" towards its goal in an optimal manner.

### 4.1.3. Adapter:
The function under the integral in (33) includes a term, referred to as the "catch-all" potential, which is not associated with any clause in the Knowledge Base. Its

function is to measure unmodeled dynamic events. This monitoring function is carried out by the Adapter which implements a generic commutator principle similar to the Lie bracket discussed in section 3.1, see (24). Under this principle, if the value of the catch-all potential is empty, the current theorem statement adequately models the status of the system. On the other hand, if the theorem fails, meaning that there is a mismatch between the current statement of the theorem and system status, the catch-all potential carries the equational terms of the theorem that caused the failure. These terms are negated and conjuncted together by the Inferencer according to the commutation principle (which is itself defined by equational clauses in the Knowledge Base) and stored in the Knowledge Base as an adaptation dynamic clause. The Adapter then generates a potential symbol, which is characterized by the adaptation clause and corresponding tuning constraints. This potential is added to criterion for the theorem characterizing the interval.

The new potential symbol and tuning constraints are sent to the Planner which generates a modified local Lagrangian for the agent and goal constraint. The new theorem, thus constructed, represents adapted behavior of the system. This is the essence of reactive structural adaptation in the our model

At this point, we pause in our description to address the issue of robustness. To a large extent, the adapter mechanism of each controller agent provides the system with a generic and computationally effective means to recover from failures or unpredictable events. Theorem failures are symptoms of mismatches between what the agent thinks the system looks like and what it really looks like. The adaptation clause incorporates knowledge into the agent's Knowledge Base which represents a recovery strategy. The Inferencer, discussed next, effects this strategy as part of its normal operation.

### 4.1.4. Inferencer:

The Inferencer is an on-line equational theorem prover. The class of theorems it can prove are represented by statements of the form of (33) and (34), expressed by an existentially quantified conjunction of equational terms of the form:

$$\exists Z \big( W_1(Z,p) \; rel_i \; V_1(Z,p) \wedge \ldots \wedge W_n(Z,p) \; rel_i \; V_n(Z,p) \big) \qquad (35)$$

where $Z$ is a tuple of variables each taking values in the domain $D$, $p$ is a list of parameters in $D$, and $\{W_i, V_i\}$ are polynomial terms in the semiring polynomial algebra:

$$\widetilde{D}\langle \Omega \rangle \qquad (36)$$

where $\widetilde{D} = (D, \langle +, \cdot, 1, 0 \rangle)$ a semiring algebra with additive unit 0 and multiplicative unit 1. In (35), $rel_i$, $i = 1, \ldots, n$ are binary relations on the polynomial algebra. Each $rel_i$ can be either an equality relation ($=$), inequality relation ($\neq$), or a partial order relation ($<$). In a given theorem, more than one partial order relation may appear. In each theorem, at least one of the terms is a partial order relation that defines a complete lattice on the algebra that corresponds to the optimization problem. This lattice has a minimum element if the optimization problem has a minimum. Given a theorem statement of the form of (35) and a knowledge base of equational clauses, the Inferencer determines whether the statement logically follows from the clauses

in the Knowledge Base, and if so, as a side effect of the proof, generates a non-empty subset of tuples with entries in $M$ giving values to $Z$. These entries determine the agent's actions. Thus, a side effect is instantiation of the agent's decision variables. In (36), $\Omega$ is a set of primitive unary operations, $\{v_i\}$, the infinitesimal primitive fields defined in section 3. Each $v_i$ maps the semiring algebra, whose members are power series involving the composition of operators from $Z$ to itself

$$v_i : \widetilde{D}\langle\langle Z \rangle\rangle \to \widetilde{D}\langle\langle Z \rangle\rangle. \tag{37}$$

These operators are characterized by axioms in the Knowledge Base and are process dependent. In formal logic, the implemented inference principle can be stated as follows: Let $\Sigma$ be the set of clauses in the Knowledge Base. Let $\Rightarrow$ represent implication. Proving the theorem means to show that it logically follows from $\Sigma$, i.e.

$$\Sigma \Rightarrow theorem. \tag{38}$$

The proof is accomplished by sequences of applications of the following inference axioms:

**(i)** equality axioms
**(ii)** inequality axioms
**(iii)** partial order axioms
**(iv)** compatibility axioms
**(v)** convergence axioms
**(vi)** knowledge base axioms
**(vii)** limit axioms

The specifics of these inference axioms can be found in [13] where it is shown that each of the inference principles can be expressed as an operator on the Cartesian product

$$\widetilde{D}\langle\langle W \rangle\rangle \times \widetilde{D}\langle\langle W \rangle\rangle. \tag{39}$$

Each inference operator transforms a relational term into another relational term. The inferencer applies sequences of inference operators on the equational terms of the theorem until these terms are reduced to either a set of ground equations of the form of (40) or it determines that no such ground form exists.

$$Z_i = \alpha_i, \alpha_i \in D \tag{40}$$

The mechanism by which the inferencer carries out the procedure described above is by building a procedure for variable goal instantiation which in our case is a locally finite automaton. We refer to this automaton as the Proof Automaton. This important feature is unique to our approach. The proof procedure is customized to the particular theorem statement and Knowledge Base instance it is currently handling. The structure of the proof automaton generated by the Inferencer is illustrated in Figure 8.

In Figure 8, the initial state represents the equations associated with the theorem. In general, each state corresponds to a derived equational form of the theorem
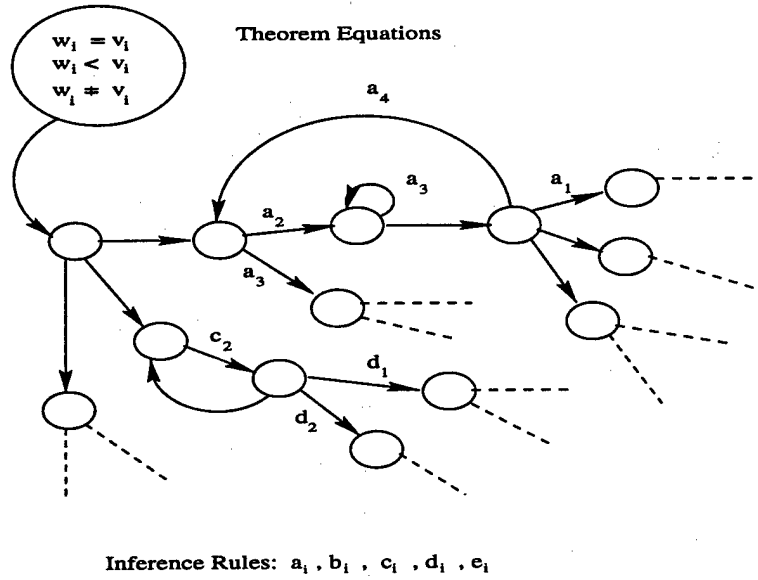
**Fig. 8.** Conceptual Structure of the Proof Automaton

through the application of a chain of inference operators to the initial state that is represented by the path,

$$S_0 \xrightarrow{inf_1} S_1 \xrightarrow{inf_2} \cdots \xrightarrow{inf_k} S_k.$$

Each edge in the automaton corresponds to one of the possible inferences. A state is terminal if its equational form is a tautology, or it corresponds to a canonical form whose solution form is stored in the Knowledge Base. In traversing the automaton state graph, values or expressions are assigned to the variables. In a terminal state, the equational terms are all ground states (see (40)). If the automaton contains at least one path starting in the initial state and ending in a terminal state, then the theorem is true with respect to the given Knowledge Base and the resulting variable instantiation is a valid one. If this is not the case, the theorem is false. The function of the complete partial order term present in the conjunction of each theorem provable by the inferencer is to provide a guide for constructing the proof automaton. This is done by transforming the equational terms of the theorem into a canonical fixed point equation, called the Kleene-Schutzenberger Equation (KSE) [13], which constitutes a blueprint for the construction of the proof automaton. This fixed point coincides with the solution of the optimization problem formulated in (33) (34), when it has a solution. The general form of KSE is

$$Z = E(p) \cdot Z + T(p) \tag{41}$$

In (41), $E$ is a square matrix, with each entry a rational form constructed from the basis of inference operators described above, and $T$ is a vector of equational forms from the Knowledge Base. Each non-empty entry, $E_{i,j}$, in $E$ corresponds to

the edge in the proof automaton connecting states $i$ and $j$. The binary operator "·" between $E(p)$ and $Z$ represents the "apply inference to" operator. Terminal states are determined by the non-empty terms of $T$. The $p$ terms are custom parameter values in the inference operator terms in $E(\cdot)$.

A summary of the procedure executed by the inferencer is presented in Figure 9.
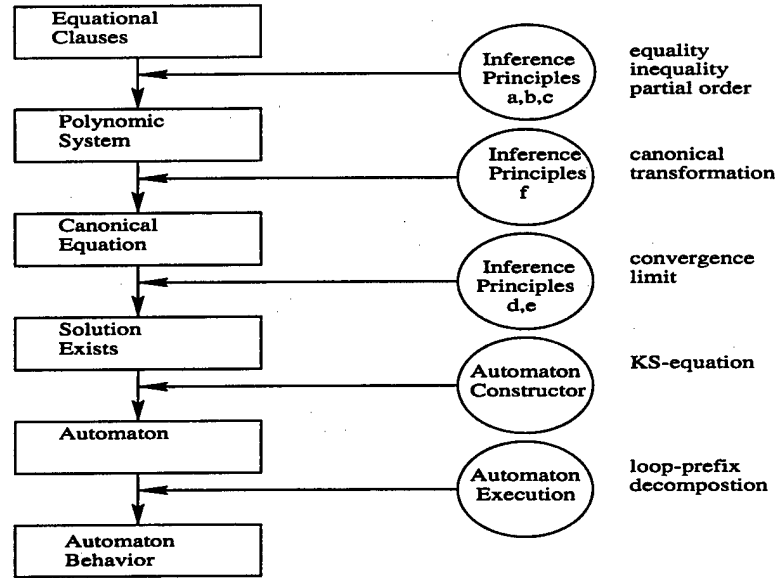


**Fig. 9.** Summary of Inferencer Procedure

We note that the construction of the automaton is carried out from the canonical equation and not by a non-deterministic application of the inference rules. This approach reduces the computational complexity of the canonical equation (low polynomic) and is far better than applying the inference rules directly (exponential).

The automaton is simulated to generate instances of the state, action and evaluation variables using an automaton decomposition procedure [41] which requires $n\log_2 n$ time, where $n$ is the number of states of the automaton. This "divide and conquer" procedure implements the recursive decomposition of the automaton into a cascade of parallel unitary (one initial and one terminal state) automata. Each of the resulting automata on this decomposition is executed independently of the others. The behavior of the resulting network of automata is identical with the behavior obtained from the original automaton, but with feasible time complexity.

The Inferencer for each control agent fulfills two functions, (i) to generate a proof for the system behavior theorem of each agent generated by the Planner (equations (33) and (34)) and (ii) to function as the central element in the Knowledge Decoder. We now describe its function for proving the behavior theorem. Later, we will overview its function as part of the Knowledge Decoder. To show how the Inferencer is used to prove the Planner theorem, (33), (34), first, we show how this theorem is

transformed into a pattern of the form of (35). Since (33), (34) formulates a convex optimization problem, a necessary and sufficient condition for optimality is provided by the following dynamic programming formulation.

$$V_i(Y,\tau) = inf_{\alpha_i} \int_\tau L_i(\Psi_i(\tau, Y), v_i|_p(G_i(\tau, p)))d\alpha(p, d\tau) \qquad (42)$$

$$\frac{\partial V_i}{\partial \tau} = inf_{\alpha_i} H(Y, \frac{\partial V_i}{\partial \tau}, \alpha_i)$$

$$\text{where } Y(t) = p \text{ and } \tau \in [t, t + \Delta)$$

In (42), the function $V_i$, called the optimal cost-to-go function, characterizes minimality starting from any arbitrary point inside the current interval. The second equation is the corresponding Hamilton-Jacobi-Bellman equation for the problem stated in (33) and (34) where $H$ is the Hamiltonian of the relaxed problem. This formulation provides the formal coupling between deductive theorem proving and optimal control theory. The inferencer allows the real-time optimal solution of the formal control problem resulting in intelligent distributed real-time control of the multiple-agent system. The central idea for inferring a solution to (42) is to expand the cost-to-go function $V(.,.)$ in a rational power series $V$ in the algebra

$$\tilde{D}\langle\langle(Y,\tau)\rangle\rangle. \qquad (43)$$

Replacing $V$ for $V_j$ in the second equation in (42) gives two items: a set of polynomic equations for the coefficients of V and a partial order expression for representing the optimality. Because of convexity and rationality of $V$, the number of equations to characterize the coefficients of V is finite. The resulting string of conjunctions of coefficient equations and the optimality partial order expression are in the form of (35). A detailed algorithmic approach to solving (42) which we call hybrid dynamic programming can be found in [28]

In summary, for each agent, the inferencer operates according to the following procedure.

**Step 1:** Load current theorem (33), (34).
**Step 2:** Transform theorem to equational form (35) via (42).
**Step 3:** Execute proof according to figure 9.

If the theorem logically follows from the Knowledge Base (i.e., it is true), the inferencer procedure will terminate on step 3 with actions. If the theorem does not logically follow from the Knowledge Base, the Adapter is activated, and the theorem is modified by the Planner according to the strategy outlined above. This mechanism is the essence of reactivity in the agent. Because of relaxation and convexity, this mechanism ensures that the controllable set of the domain is strictly larger than the mechanism without this correction strategy.

### 4.1.5 Knowledge Decoder:

The function of the Knowledge Decoder is to translate knowledge data from the network into the agent's Knowledge Base by updating the inter-agent specification clauses. These clauses characterize the second constraint in (42). Specifically, they

express the constraints imposed by the rest of the network on each agent. They also characterize the global-to-local transformations (see [23]). Finally, they provide the rules for building generalized multipliers for incorporating the inter-agent constraints into a complete unconstrained criterion, which is then used to build the cost-to-go function in the first expression in (42). A generalized multiplier is an operator that transforms a constraint into a potential term. This potential is then incorporated into the original Lagrangian of the agent which now accounts explicitly for the constraint.

The Knowledge Decoder has a built-in inferencer used to infer the structure of the multiplier and transformations by a procedure similar to the one described for (14). Specifically, the multiplier and transformations are expanded in a rational power series in the algebra defined in (43). Then the necessary conditions for duality are used to determine the conjunctions of equational forms and a partial order expression needed to construct a theorem of the form of (42) whose proof generates a multiplier for adjoining the constraint to the Lagrangian of the agent as another potential.

The conjunction of equational forms for each global-to-local transformation is constructed by applying the following invariant imbedding principle:

> For each agent, the actions at given time t in the current interval, as computed according to (42), are the same actions computed at t when the formulation is expanded to include the previous, current, and next intervals.

By transitivity and convexity of the criterion, the principle can be analytically extended to the entire horizon. The invariant imbedding equation has the same structure as the dynamic programming equation given in (42), but with the global criterion and global Hamiltonians instead of the corresponding local ones.

The local-to-global transformations are obtained by inverting the global-to-local transformations which, in turn, are obtained by expressing the invariant embedding equation as an equational theorem of the form of (35). These inverses exist because of convexity of the relaxed Lagrangian and the rationality of the power series.

It is important at this point to interpret the functionality of the Knowledge Decoder of each agent in terms of what it does. The multiplier described above has the effect of aggregating the rest of the system and the other agents into an equivalent companion system and companion agent, respectively, as viewed by the current agent. This is illustrated in Figure 10.

The aggregation model (Figure 10) describes how each agent perceives the rest of the network. This unique feature allows us to characterize the scalability of the architecture in a unique manner. Namely in order to determine computational complexity of an application, we have only to consider the agent with the highest complexity (i.e., the local agent with the most complex criterion) and its companion.

## 4.2 Architectural Elements of a Declarative Control Network

The inter-agent communication network's main function is to transfer inter-agent constraints among agents according to a protocol written in equational Horn clause language. These constraints include application dependent data and, most importantly, inter-agent synchronization. The inter-agent synchronization strategy is very simple. An agent is synchronous with respect to the network if its inter-agent constraint multiplier is continuous with respect to the current instance of its active
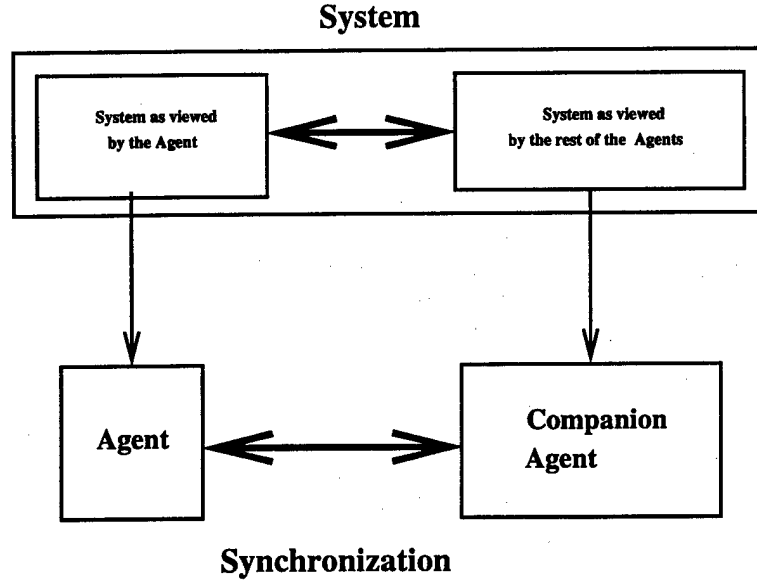
**System**



**Synchronization**

**Fig. 10.** The Companion Agent

knowledge. Since the equational Horn clause format allows for the effective test of continuity (which is implicitly carried out by the inferencer in the Knowledge Decoder), a failure in the Knowledge Decoder theorem results in a corrective action toward re-establishing continuity with respect to the topology defined by the current instance of the Knowledge Base [20].

The specification of the geometry of the network, as a function of time, is dictated primarily by global observability. By global observability, we mean the closure of the knowledge of the system as whole relative to the scope the systems reactivity. One of the central tasks in any application is to provide knowledge in the equational clause format to characterize global observability for the hybrid system.

### 4.3 Summary the MAHCA Architecture

Our formulation gives a precise statement of a hybrid control problem in terms of a multiple agent hybrid declarative control. Our approach characterizes the problem via a knowledge base of equational rules that describes the dynamics, constraints and requirements of plant.

For the traffic control application, the Behavior statement or ATC statement is represented as a Lagrangian of the form

$$min_\alpha \int_\Omega L(\rho^c, v^c, \rho^v, v^v, \frac{\partial \rho^c}{\partial x}, \frac{\partial v^c}{\partial x}, \frac{\partial \rho^v}{\partial x}, \frac{\partial v^v}{\partial x}, \frac{\partial \rho^c}{\partial t}, \frac{\partial v^c}{\partial t}, \frac{\partial \rho^v}{\partial t}, \frac{\partial v^v}{\partial t}, t, x) d\alpha \quad (44)$$

where $\Omega = \{\langle x, t \rangle : 0 \leq t \leq \infty, 0 \leq x \leq 2.8\}$

The Goal is maximize the average throughput, given as:

$$max\left(lim_{T\to\infty}\int_0^T\int_0^L Q(\rho^c,v^c,\rho^v,v^v,V^u)dxdt.\right) \tag{45}$$

The sensor data is given by

$$\rho^c(\tau,x_i) \quad v^c(\tau,x_i)$$
$$\rho^v(\tau,x_i) \quad v^v(\tau,x_i)$$

where $i = 1,2,\ldots,9$ to represent sensor inputs at one-third mile intervals and $\tau = 1,2,\ldots$ seconds.

Finally, the actions for the traffic control consist solely of the commanded velocity vector $V^u$.

We have developed a canonical representation of interacting networks of controllers. Given a connectivity graph with N nodes (controllers) and the corresponding agent's knowledge bases, a network of 2N agents can be constructed with the same input-output characteristics, so that each agent interacts only with another (equivalent) companion agent, whose knowledge base is an abstraction of the knowledge in the network. Thus, in general, the multiple-agent controller for any network configuration is reduced to a set of agent pairs.

One agent of the agent pair maintains coordination with other agent pairs across the network. We call that agent of the pair which represents network information the Thevenin Agent, after the author of a similar theorem in electrical network theory. The proof carried out by the Thevennin Agent generates, as a side effect, coordination rules that define what and how often to communicate with other agents. These rules also define what the controller needs from the network to maintain intelligent control of its physical plant.

Our approach develops a canonical way to prove the theorem characterizing the desired behavior for each agent by constructing and executing on-line a finite state machine called the "proof automaton." The inference process is represented as a recursive variational problem in which the criterion is an integral of a function called the Generalized Lagrangian. The Generalized Lagrangian maps the Cartesian product of equational rules and inference principles to the real line, thus effectively providing a hill-climbing heuristic for the inference strategy of the theorem prover (see section 3). In MAHCA the inference steps play a role analogous to action signals in conventional control while the vector fields on the carrier manifold $M$ constitute generators of feedback laws (section 2).

## 5 Simulation

### 5.1 Simulation Parameters and Operation

The simulation parameters and constants for the demonstration runs were

$T$ Length of simulation run = 20 minutes, with data snapshots of 20 second intervals (fixed).

$k_3$ Percentage of cars complying with commanded velocity varied from 5% to 9%. This rule was not rigid. The simulation runs demonstrated adaption of this rule (to a lower value) when the system began drifting to an instable condition.

$\delta$ Average response time = 1.5 seconds (variable).

$C^c$ Car capacity coefficient, determined for a 4 lane freeway.

Those parameters marked "variable" were subject to adaptation by the architecture in response to explicit or inferred changes in the environment.

Accidents were introduced at a random point (after a 100 second "priming" period) at a random location between freeway position $x = 1.38$ and $x = 1.43$. At this point $C^c$ was reduced by one-half.

The simulation uses a standard 4-th order Runga-Kutta Integration scheme for time and 2-nd order Adam-Bashworth integration for space.

The simulation produces loop sensor values as inputs for the control functions. Sensors appear in pairs (120 ft apart) at one-third mile intervals and provide velocity and density data for vehicles and voids. The simulation simulates ideal sensors, free of bad data and able to detect voids.

The simulation was implemented in Prolog using the same set of equations as those encoded into the Knowledge Base. Although the equations are the same, the simulation is completely different operational entity and only interacts with the MAHCA control agents by providing them with sensor inputs and accepting their control actions.

Execution of the simulation is performed by a single top-level prolog rule.

$$traffic\_demo(Inputs, Outputs) :-$$
$$simulation\_on\_off(Stop),$$
$$simulation(Inputs, Sensor\_out, Action\_in, Outputs),$$
$$controller\_on\_off(Bool), \qquad\qquad (46)$$
$$record(Actions\_in, Sensors\_out, Inputs, Outputs),$$
$$repeat.$$

In this rule, *traffic_demo* is the main rule. The *Inputs* parameter is expected to contain the operational parameters (such as maximum running time). The *Outputs* parameter will become instantiated with the status of the simuluation.

The *simulation_on_off* rule provides a mechanism for setting *Stop* to true to shut down the simulation.

In *Simulation*, the *Inputs* and *Action_in* parameters are used as inputs to compute the next set of *Sensor_out* values and a new set of simulation *Outputs* (used for recording purposes only).

Like *simulation_on_off*, *contoller_on_off* is a mechanism for setting *Bool* to *True* to invoke the controller or *False* to turn it off. During the initial 100 second priming period, *Bool* will be *False*.

The *controller* rule invokes the inferencing process to compute a new set of *Action_in* values for the simulator from the set of *Sensor_out* values.

Finally, *record* will report, to the terminal and/or to the data files, the values from the action lists, sensor outputs, and other simulation and controller inputs and

outputs. The data was recorded every 60 seconds, starting with $t = 160$, the end of the first control period following the initial 100 second priming.

## 5.2 Discussion

The control will compute and broadcast travel speeds for each segment of the freeway. When followed by individuals, the throughput of the whole system will increase. Compliance on the part of drivers assumes adoption of the philosophy that adherence to less than intuitive speeds is in their best interest.

The control can also be used to determine and transmit navigational instructions throughout a network of freeway corridors, accomplished by tracking the wave propagations over the segments and guiding a percentage of the vehicles to maximize throughput of the system as opposed to minimizing travel for individual cars.

In the abstraction of individual vehicles into a continuous wave model, some information is lost. Since the freeway segment is regarded as one unit, with a width parameter representing the number of lanes, knowledge of any particular lane is not tracked. Incidences, real, such as stalled vehicles, or imaginary, such as "rubber necking", are modeled as a reduction in the width of the segment. Individual vehicles are simply contributing particles to the wave, so exact speed and location of any one vehicle is not known.

This model can easily be extended to perform other duties for an intelligent highway system. For example, detection of incidents can be accomplished by inspecting shock waves which result for such discontinuities. From the characteristic of the shock wave, the nature of the incident can be determined, e.g. a slow moving truck or speeding emergency vehicle. This capability is theoretically possible and has been demonstrated in a simulation, but it is very difficult to validate with empirical data. By comparing inconsistencies between the observed and expected density and velocity values in the waves as they propagate along the freeway corridor, errant sensor data can be eliminated. For traffic signal processing, red lights can be viewed as a total restriction of the highway and green lights as an instantaneous removal of the restriction which is naturally coordinated with signal changes for yellow lights and cross traffic. The modeling of voids becomes especially important in these cases.

The reactive capability of the execution strategy of the architecture is exploited to provide two major functions. Appearance of shock waves in the solution triggers reactivity to decompose the wave generator into continuous and shock components to reflect changed conditions. Over the longer term, the reactivity can be used to fine tune the equations to account for subtle changes in the nature of the vehicles, such as average length, and driver policies, such as speeds and following distances.

## 5.3 Preliminary Analysis

This section gives a brief preliminary analysis of the results of simulations that were run by Brian Coles of Intermetrics. The referenced charts appear at the end of the paper. There are three sets of charts, one with 5 % of controlled cars, one with 8 % of controlled cars, and one with 9 % of controlled cars. (With this last set, the 9 % represents the initial percentage of controlled cars. Over time, this was lowered by the Inferencer.) In each set there are three graphs showing the densities and

velocities of cars and voids, taken at $t = 160$, $t = 700$, and $t = 1300$, the end of the simulation run. The last two graphs in each set show the differences in velocity and density values over intervals equal to about 1/4 of the simulation run. For all of these graphs, the horizontal units are miles along the corridor. Velocities are measured in miles/hour and densities are measured in cars (or voids) per mile.

Along the portion of the freeway corridor before the accident, the density wave initially has a high amplitude, but as the velocity control of the cars begins to take effect, the amplitude of the wave diminishes. Charts 1-3, 6-8, and 11-13 show this phenomena for samples of freeway dynamics at different times and for different percentages of control.

After the accident along the freeway, control is not needed because vehicle densities never approach the saturation limit. Vehicles are free to travel as fast as they want, within legal limits, of course.

For 9% control, versus no control, the throughput increases 28%. Less than 5% control yields unstable behavior and is not effective in eliminating grid-lock. As the percentage of controlled cars increases between 5% and 9%, the grid-lock is monotonically reduced, compare charts 3 and 13. At 9% control, the grid-lock is successfully cleared. Between 9% and 14%, the gridlock is successfully cleared, but throughput is reduced. In an effort to achieve the goal of maximizing throughput, the controller reduces the percentage of control vehicles. Above 14% control, the system experiences performance degradation due to overdeterminism.

The interaction between void and car density dynamics shown in charts 4, 5, 9, 10, 14, and 15 demonstrates many density distributions that are possible, but can not be generated by representing the dynamics as car densities alone.

The wave segments generated in the controller allow for easy decomposition between continuous and shock elements, see for example charts 1 and 6. This capability alone allows the control to infer accidents and recompute the control law accordingly. We observed a one-to-one correlation (as expected) between theorem failures and the appearance of the shock component.

In the accident region, the four variables show peaks, see charts 2, 7, 12. These peaks, however, do not appear at the same freeway locations. There are phase shifts which are due to the asymmetry between the car and void particles. The void velocities has a phase shift over car velocities, see charts 4, 5, 9, 10, 14 and 15. This phase shift is expoited by the controller to schedule the commanded velocity profile to meet the goal of maximizing throughput. The effects of the dynamics due to an incident are detected earliest in the void density dynamics.

Void and car densities together appear to satisfy the additive conservation law, as to be expected from models for binary population dynamics (void and car particles).

The flow of velocity and density waves qualitatively agrees with emperical observation.

Our runs show that traffic congestion latency decreases significantly as the percentage of controlled cars increases. After incident removal with no control, the car density goes to nominal after 32 minutes. With 9% control the car density goes to nominal after only 12 minutes. This difference is directly due to the scheduling of velocities before the accident. That is, the command velocity distribution before the accident under control is lower than the velocity distribution with no control

# References

1. Alur, R., Henzinger, T.A., and Sontag, E.D., eds., *Hybrid Systems III*, Lecture Notes in Computer Science vol. 1066, Springer-Verlag, (1996).

2. Antsaklis, P., Kohn, W., Nerode, A, and Sastry, S. eds., *Hybrid Systems II*, Lecture Notes in Computer Science vol. 999, Springer-Verlag, (1995).

3. Crossley, J.N., Remmel, J.B., Shore, R.A. and Sweedler, M.E., *Logical Methods* Birkhauser, (1993).

4. Dodhiawala, R.T., V. Jagoenathan and L.S. Baum, "Erasmus System Design: Performance Issues" Proceedings of Workshop on Blackboard Systems Implementation Issues, AAAI, Seattle, WA., July 1987.

5. Garcia, H.E. and A. Ray "Nonlinear Reinforcement Schemes for Learning Automata" Proceedings of the 29th IEEE CDC Conference, Vol. 4, pp 2204- 2207, Honolulu, HA, Dec. 5-7, 1990.

6. Ge, X., Kohn, W., Nerode, A. and Remmel, J.B., "Algorithms for Chattering Approximations to Relaxed Optimal Control. MSI Tech. Report 95-1, Cornell University. (1995)

7. Ge, X., Kohn, W., Nerode, A. and Remmel, J.B.,"Hybrid Systems: Chattering Approximations to Relaxed Control," *Hybrid Systems III*, (R. Alur, T.A. Henzinger, E.D. Sontag, eds.) Lecture Notes in Computer Science **1066**, Springer, (1996), 76-100.

8. Gelfand, I.M. and Fomin, S.V., *Calculus of Variations*, Prentice Hall, 1963.

9. Grossman, R.L., Nerode, A., Ravn, A. and Rischel, H. eds., *Hybrid Systems*, Lecture Notes in Computer Science 736, Springer-Verlag, (1993).

10. Kohn, W., "A Declarative Theory for Rational Controllers" Proceedings of the 27th IEEE CDC, Vol. 1, pp 131-136, Dec. 7-9, 1988, Austin, TX.

11. Kohn, W., "Application of Declarative Hierarchical Methodology for the Flight Telerobotic Servicer" Boeing Document G-6630-061, Final Report of NASA- Ames research service request 2072, Job Order T1988, Jan. 15, 1988.

12. Kohn, W., "Rational Algebras; a Constructive Approach" IR&D BE-499, Technical Document D-905-10107-2, July 7, 1989.

13. Kohn, W., "The Rational Tree Machine: Technical Description & Mathematical Foundations" IR&D BE-499, Technical Document D-905-10107-1, July 7, 1989.

14. Kohn, W., "Declarative Hierarchical Controllers" Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, pp 141-163, Pacifica, CA, July 17-19, 1990.

15. Kohn, W., "Declarative Multiplexed Rational Controllers" Proceedings of the 5th IEEE International Symposium on Intelligent Control, pp 794-803, Philadelphia, PA, Sept. 5, 1990.

16. Kohn, W., "Declarative Control Architecture" CACM Aug 1991,Vol34, No8.

17. Kohn, W., "Advanced Architectures and Methods for Knowledge-Based Planning and Declarative Control" IR&D BCS-021, ISMIS'91, Oct. 1991.

18. Kohn W., "Multiple Agent Inference in Equational Domains Via Infinitesimal Operators" Proc. Application Specific Symbolic Techniques in High Performance Computing Environment". The Fields Institute, Oct 17-20 1993.

19. Kohn W., "Multiple Agent Hybrid Control" Proc of the NASA-ARO Workshop on formal Models for Intelligent Control, MIT,sept 30- Oct2, 1193.

20. Kohn, W. and Murphy, A., "Multiple Agent Reactive Shop Floor Control" ISMIS'91, Oct. 1991.

21. Kohn, W. and Nerode, A., "Multiple Agent Declarative Control Architecture" Proc. of the workshop on Hybrid Systems, Lygby, Denmark, Oct 19-21, 1992.

22. Kohn, W. and Nerode, A., "Multiple Agent Hybrid Control Architecture" In [9], (1993), 297-316.

23. Kohn W., and Nerode, A., "Multiple-Agent Hybrid Systems" Proc. IEEE CDC 1992, vol 4, pp 2956-2972.

24. Kohn, W. and Nerode, A., "An Autonomous Systems Control Theory: An Overview" Proc. IEEE CACSD'92, March 17-19, Napa, Ca.,pp 200- 220.

25. Kohn W,, and Nerode A. "Models For Hybrid Systems: Automata, Topologies, Controllability, Observability" In [9], (1993) 317-356.

26. Kohn W. and Nerode, A., "Multiple Agent Autonomous Control-A Hybrid Systems Architecture" In [3], (1993) 593-623.

27. Kohn, W., Nerode, A. and Remmel, J.B., " Hybrid Systems as Finsler Manifolds: Finite State Control as Approximation to Connections", In [2], (1995), 294-321.

28. Kohn, W., Nerode, A. and Remmel, J.B., "Feedback Derivations: Near Optimal Controls for Hybrid Systems", Proceedings of CESA'96 IMACS Multiconference, Vol 2. 517-521.

29. Kohn, W., Nerode, A. and Remmel, J.B., "Continualization: A Hybrid Systems Control Technique for Computing", Proceedings of CESA'96 IMACS Multiconference, Vol 2. 507-511.

30. Kohn, W. and Remmel, J.B., "Digital to Hybrid Program Transformations", Proceedings of the 1996 IEEE International Symposium on Intelligent Control, 342-347.

31. Kohn, W., and Remmel, J.B., "Implementing Sensor Fusion Using a Cost Based Approach", to appear in the Proceedings of ACC'97.

32. Kohn,W., Remmel, J.B., and Nerode, A. Scalable Data and Sensor Fusion via Multiple-Agent Hybrid Systems, submitted to IEEE Transactions on Automatic Control.

33. Kohn, W. and T. Skillman, "Hierarchical Control Systems for Autonomous Space Robots" Proceedings of AIAA Conference in Guidance, Navigation and Control, Vol. 1, pp 382-390, Minneapolis, MN, Aug. 15-18, 1988.

34. Kowalski, R., *Logic for Problem Solving*, North Holland, NY, 1979.

35. Kuich, W. and Salomaa, A., *Semirings, Automata, Languages*, Springer Verlag, NY., 1985.

36. Lloyd, J.W. *Foundations of Logic Programming* second extended edition, Springer Verlag, NY, 1987.

37. Liu, J.W.S., "Real-Time Responsiveness in Distributed Operating Systems and Databases" proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, Pacifica, CA., July 17-19, 1990, pp 185-192.

38. Nii, P.H., "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures" the AI Magazine, Vol. 7, No. 2, Summer 1986, pp 38-53.

39. Padawitz, P., '*Computing in Horn Clause Theories*, Springer Verlag, NY, 1988.

40. Robinson, J.A., '*Logic: Form and Function*, North Holland, NY, 1979.

41. Skillman, T. and Kohn, W., et.al., "Class of Hierarchical Controllers and their Blackboard Implementations" Journal of Guidance Control & Dynamics, Vol. 13, N1, pp 176-182, Jan.-Feb., 1990.

42. Warner, F.W., *Foundations of Differential Manifolds and Lie Groups*, Scott-Foresman, Glenview, Ill.

43. Warga, K., '*Optimal Control of Differential and Functional Equations*, Academic Press, NY., 1977.

44. Witham, M., *Wave Dynamics: Theory and Application*, McGraw Hill, (1976).

45. Young, L.C., '*Optimal Control Theory*, Chelsea Publishing Co., NY, 1980.